Let $N$ be the class of functions realizable by feedforward linear threshold nets with $n$ input units, two hidden units each of zero threshold, and an output unit. This class is also essentially equivalent to the class of intersections of two open half spaces that are bounded by planes through the origin. We give an algorithm that probably almost correctly (PAC) learns this class from examples and membership queries. The algorithm runs in time polynomial in $n$, $\epsilon$ (the accuracy parameter), and $\delta$ (the confidence parameter). If only examples are allowed, but not membership queries, we give an algorithm that learns $N$ in polynomial time provided that the probability distribution $D$ from which examples are chosen satisfies $D(x) = D(-x) \forall x$. The algorithm yields a hypothesis net with two hidden units, one linear threshold and the other quadratic threshold.

1 Introduction

The perceptron algorithm (Rosenblatt 1962) is the launching point for much of modern neural network research. This algorithm provably finds a classifier for any set of linearly separable examples, and thus has applications, for example, to pattern recognition tasks. Unfortunately, as was stressed by Minsky and Papert (1969), the perceptron algorithm does not work in polynomial time. However, Khachian (1979) and also Karmarkar (1984) have provided algorithms for classifying linearly separable examples that are polynomial time. Blumer et al. (1987) subsequently showed how such polynomial time classification algorithms could be employed to produce polynomial time PAC learning algorithms.

Minsky and Papert (1969) also stressed that linear separability is too strong a condition to expect, and thus learning algorithms must deal with more complex target functions to be practical. Thus attention is now focused on networks with hidden units. The main learning heuristic is backpropagation (Le Cun 1986; Rumelhart et al. 1986), and a major question is whether there is any algorithm for training networks with hidden units that scales well, that is, will work rapidly for large networks.
The simplest such networks are the sort we will study in this paper, which have two hidden units each of threshold zero (see Fig. 1). The learning problem for such networks is equivalent (as we will remark) to learning functions described by intersections of half spaces. There has been intensive work on this problem (see, e.g., Ridgeway 1962; Blumer et al. 1989; Baum 1990a).

A negative partial answer was provided by Blum and Rivest (1988) who showed that there is no polynomial time algorithm that can solve the loading problem for networks with two hidden units. This obstruction can be evaded if we consider more flexible algorithms (see Baum 1989 for a review). Their result means it is unlikely we will find an algorithm that trains neural networks to their full potential. However, even if we could train a network with two hidden units to its potential, it could only learn target functions that can be represented by such a network. We might just as well train a larger network to realize such target functions. In this paper we give an algorithm that provably learns such functions, and produces as its output a feedforward net with two hidden units; however, one of these two units is a quadratic threshold unit, rather than linear threshold.
We work first within the PAC learning model (Valiant 1984). In this model a class $C$ of boolean functions is called learnable if there is an algorithm $A$ and a polynomial $p$ such that for every $n$, for every probability distribution $D$ on $\mathbb{R}^n$, for every $c \in C$, for every $0 < \epsilon, \delta < 1$, $A$ calls examples and with probability at least $1 - \delta$ supplies in time bounded by $p(n, s, \epsilon^{-1}, \delta^{-1})$ a function $g$ such that

$$\text{Prob}_{x \in D}[c(x) \neq g(x)] < \epsilon$$

Here $s$ is the "size" of $c$, that is, the number of bits necessary to encode $c$ in some "reasonable" encoding.

This model thus allows the algorithm $A$ to see classified examples drawn from some natural distribution and requires that $A$ output a hypothesis function which with high confidence ($1 - \delta$) will make no more than a fraction $\epsilon$ of errors on test examples drawn from the same natural distribution. This model thus corresponds reasonably well to what is often desired in practice, for example, in applications of backpropagation.

Let $N$ denote the class of boolean functions defined by feedforward nets of linear threshold units with two hidden units, both of threshold zero. We will supply for this class an algorithm fulfilling the above conditions provided that one restrict to probability distributions with an inversion symmetry: $D(x) = D(-x)$. We will say a class $C$ is $i$-learnable in this case (i.e., if we supply an $A$ that satisfies the above conditions when we restrict to inversion symmetric distributions).

The inversion symmetric condition is fulfilled for some natural distributions, for example, for the uniform distribution on the unit sphere, $S^n$. It is evidently not likely to hold exactly in practice. Although our proof will fail if this condition is violated, it is plausible (and an interesting open problem) that related methods might be robust to small violations of this symmetry. We will return to this point in Section 3.

A natural way to increase the power of the learner is to allow, in addition to examples, membership queries (Valiant 1984). Here the algorithm $A$ supplies an $x$, and is told the classification $c(x)$. This protocol is thus appropriate when we have a teacher who can answer questions, for example, a human expert, or when we can experiment. We believe learning with membership queries is far closer to the way people learn than simply through examples, and that it is quite a reasonable extension in practical applications, for example, in optical character recognition or speech recognition, where a human expert could classify examples posed by a learning algorithm. This protocol has also been studied (see, e.g., Angluin 1988), but it has been largely overlooked in the neural net literature. We will give here an algorithm that in polynomial time learns $N$ for arbitrary distributions when membership queries are allowed. A separate publication (Baum 1990b) will describe more powerful algorithms that are able to learn from examples and queries, in polynomial time, substantially wider classes than $N$. 
Polynomial Time Algorithm

2 Preparatory Remarks on PAC Learning

We will make use of the following theorem.

Theorem 1. (Blumer et al. 1988). Let $C$ be a well behaved concept class of VC dimension $d$. Then if we call

$$M_0(\epsilon, \delta, d) = \max \left( \frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{8d}{\epsilon} \log \frac{13}{\epsilon} \right)$$

examples $(x, t)$ from any distribution $D'$ on $\mathbb{R}^n \times (0,1)$ and find a $c \in C$ consistent with these examples (i.e., $c(x) = t(x)$ for all these examples) then we have confidence $1 - \delta$ that

$$\text{Prob}_{(x,t) \in D'}[c(x) \neq t(x)] < \epsilon$$

We must next introduce a specification of the size of our target functions, which will depend on the number of bits of precision $a$ with which we work. The reason why we need such a notion is the following. Say we wished to learn target functions specified by a single half plane. If an adversary were allowed to cluster the probability distribution $D$ arbitrarily close to the plane, then we might need an arbitrary amount of time to find a plane separating positive from negative examples, since, for example, Khachian's or Karmarkar's algorithms take time depending polynomially on the number of bits of precision with which one works. One could avoid this problem by imposing a continuity condition on the probability distribution $D$, or by restricting the accuracy with which the target function is defined. Instead we will adopt the simplest solution for our case, which is to assume that the examples lie on a lattice of spacing $2^{-a}$, that is, for any example $x \in 2^{-a} \mathbb{Z}^n$, and $D(x) = 0$ for $\|x\| > 1$. With this assumption, Karmarkar's algorithm will be able to find linear separators in time polynomial in $a, n,$ and the number of examples being separated.

3 Learning from Examples Only

We first remark that the problem of learning such feedforward nets is trivially equivalent to learning an intersection of half spaces. The "neurons" in our nets are linear threshold functions: these take value 1 in a half space and zero in the complementary half space. The planes corresponding to the two hidden units thus divide the $n$-dimensional input space.
Figure 2: The hyperplanes corresponding to the two hidden units divide the input space into four regions.

space into four regions (see Fig. 2) and the value of $f(x)$ depends only on which of the four regions $x$ lies in. The position of these two planes and thus the four regions depends on the weights to the hidden units. Depending on the weights to the output unit, our net may have as positive regions any one of these regions, for example, \{a\}; any two contiguous regions, for example, \{a, b\}, but not \{a, c\}; any three regions, for example, \{a, b, c\}; or all four regions (or none). It is evident that each of these cases can be viewed as the intersection of two half spaces; or in the case of three regions, the complement of the intersection, which becomes an intersection problem if we simply reverse our definition of positive and negative examples. Thus it suffices to consider learning an intersection of two half spaces.

Let $F$ be the class of functions described as the intersection of two open half spaces on $\mathbb{R}^n$. Thus $f \in F$ may be described by giving two
vectors $w_1, w_2 \in \mathbb{R}^n$, and $f(x) = 1$ if $w_1 \cdot x > 0$ and $w_2 \cdot x > 0$, else $f(x) = 0$.\(^2\)

Let $G$ be the class of functions we call the XOR of two half spaces, and define as follows. $g \in G$ is described by giving two vectors $w_1, w_2 \in \mathbb{R}^n$, and $g(x) = 1$ if $(w_1 \cdot x)(w_2 \cdot x) > 0$, else $g(x) = 0$.

**Theorem 2.** (Blum 1989; Valiant and Warmuth 1989). $G$ is learnable.

**Proof.** Given a set $S$ of examples $[x_i, g(x_i)]$ we may find a consistent classifier by the following trick. Find a $w \in \mathbb{R}^n$ such that $\sum_{ij} w_{ij} x_i x_j > 0$ for any positive example $x \in S$ and $\sum_{ij} w_{ij} x_i x_j < 0$ for any negative example in $S$. Such a $w$ exists by the definition of $G$ since $\sum_{ij} w_{ij} x_i x_j$ is greater than or less than zero according to whether $x$ is a positive or negative example. Finding such a $w$ is a simple linear programming problem and may be accomplished using Karmarkar's algorithm (Karmarkar 1984). Now we have found a consistent classifier in the set $H$ of half spaces on $\mathbb{R}^n$, which has VC dimension $n^2 + 1$. By Theorem 1 this solves our problem if we use $M_0(\epsilon, \delta, n^2 + 1)$ examples. Q.E.D.

Now we will solve the $i$-learning problem for an intersection of two half spaces by reducing to the problem of learning XORs. We are trying to find a region $r$ defined by $w_1 \cdot x > 0, w_2 \cdot x > 0$ (see Fig. 3). Let $\hat{r}$ denote the region $w_1 \cdot x < 0, w_2 \cdot x < 0$. The idea is the following. We will find a closed half space $h$ containing very little measure for positive examples [i.e., $D(\hat{r} \cap h) \ll \epsilon$] and almost all measure in $\hat{r}$ [i.e., $D(\hat{r} \cap \mathbb{R}^n \setminus h) \ll \epsilon$]. ($\mathbb{R}^n \setminus h$ denotes the open half space which is the complement of $h$.) We may easily find such an $h$ by collecting a sufficiently large number $M_1$ of positive examples and finding a plane through the origin having all of these on one side. Theorem 1 then guarantees us that almost all positive measure is on this side. Because of the symmetry we are requiring on the distribution, if almost all measure in $r$ is on one side of this plane, almost all measure in $\hat{r}$ is on the other side. Now we may safely hypothesize that any point in $h$ is a negative example, and we have thus reduced the problem to classifying points in $\mathbb{R}^n \setminus h$. We next collect a set $S$ of examples in $\mathbb{R}^n \setminus h$. Since with very high probability, none of the points in $S$ lies in $\hat{r}$, the set $S$ will be consistent with the XOR of two planes. We will thus use the XOR algorithm. This will produce a $w \in \mathbb{R}^n$. Our hypothesis function will then be $x$ is positive if and only if $x \in h$ and $\sum_{ij} x_i x_j > 0$.

The algorithm is the following.$^3$

1. Let $M_1 = M_0(\delta \epsilon/16M_0(\epsilon, \delta/4, n^2 + 1), \delta/8, n + 1) = \tilde{O}(n^3/\epsilon^2 \delta)$. Call examples until either (a) we have $M_1$ positive examples or (b) we have called $M = \max(2M_1/\epsilon, \delta^{-1})$ examples (whichever comes first). In case (b), halt and output hypothesis $g(x) = 0$ (i.e., classify all examples as negative).

---

$^2$We restrict to hyperplanes through the origin.

$^3$For ease of notation, define $O(x)$ to mean $O(x) \times (\text{terms of logarithmic size})$. 

---

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/neco.1990.2.4.510 by guest on 30 August 2021
2. Find a half space \( h \) bounded by a hyperplane through the origin such that all the positive examples are in \( h \). This can be done by Karmarkar's algorithm.

3. Let \( S = \{ [x', f(x')] \} \) be the set of the first \( M_0(\epsilon, \delta/4, n^2 + 1) \) examples we found which are in \( h \). Find a \( w \in \mathbb{R}^n \) s.t. \( \sum_{ij} w_{ij} x_i x_j > 0 \) for any positive example \( x \in S \) and \( \sum_{ij} w_{ij} x_i x_j < 0 \) for any negative example in \( S \). \( w \) can be found, for example, by Karmarkar's algorithm.

4. Output hypothesis \( g \): \( x \) is positive if and only if \( x \in h \) and \( \sum_{ij} w_{ij} x_i x_j > 0 \).

5. Halt.

Figure 3: The input space is divided by target hyperplanes \( w_1 \) and \( w_2 \) and the hyperplane \( \partial h \), which bounds half space \( h \), that we draw in step 2 of our algorithm. Region \( r = \{ x : w_1 \cdot x \geq 0, w_2 \cdot x \geq 0 \} \) is labeled, as is \( \bar{r} = \{ x : -x \in r \} \).
Notice that the hypothesis function $g$ output by this net is equivalent to a feedforward threshold net with two hidden units. One of these is a linear threshold with output $\theta(\sum_j w_j^b x_j)$, where $w^b$ is the normal to the boundary hyperplane of $h$. The other is a quadratic threshold with output $\theta(\sum w_j x_j x_i)$. Here $\theta$ is the Heaviside threshold function and $x_i$ is the input vector.

We now prove this algorithm is correct, provided $D$ is inversion symmetric, that is, for all $x$, $D(x) = D(-x)$.

**Theorem 3.** The class $E'$ of intersections of two half spaces is $i$-learnable.

**Proof.** It is easy to see that with confidence $1 - \delta$, if step (1b) occurs, then at least a fraction $1 - \epsilon$ of all examples are negative, and the hypothesis $g(x) = 0$ is $\epsilon$-accurate. Likewise, if step (1b) does not occur, we have confidence $1 - 6/4$ that at least a fraction $\epsilon/4$ of examples are positive.

If step (1b) does not occur, we find an open half space $\mathcal{H}$ containing all the positive examples. By Theorem 1, with confidence $1 - \delta/8$, we conclude that the measure $\mu(\mathcal{H} \cap \hat{r}) < \delta \epsilon / 16 M_0(\epsilon, \delta/4, n^2 + 1)$. Now if $x \in \hat{r}$, then $-x \in \hat{r}$, and vice versa; and if $x \in \mathcal{H}$, then $-x \in \mathcal{H}$. Thus if $x \in \mathcal{H} \cap \hat{r}$ then $-x \in \mathcal{H} \cap r$. But $D(x) = D(-x)$ by hypothesis. Thus the measure $\mu(\mathcal{H} \cap \hat{r}) < \mu(\mathcal{H} \cap r)$. Now we use bound the conditional probability $\Pr(x \in \hat{r} | x \in \mathcal{H})$ that a random example in $\mathcal{H}$ is also in $\hat{r}$:

$$\Pr(x \in \hat{r} | x \in \mathcal{H}) = \frac{\mu(\mathcal{H} \cap \hat{r})}{\mu(\mathcal{H})} \leq \frac{\mu(\mathcal{H} \cap r)}{\mu(\mathcal{H})}$$

Recall we saw above that with confidence $1 - 3\delta/8$, $\mu(\mathcal{H}) > \epsilon/4$ and $\mu(\mathcal{H} \cap r) < \delta \epsilon / 16 M_0(\epsilon^{-1}, \delta/4, n^2 + 1)$. Thus we have

$$\Pr(x \in \hat{r} | x \in \mathcal{H}) < \delta / 4 M_0(\epsilon^{-1}, \delta/4, n^2 + 1)$$

Now we take $M_0(\epsilon, \delta/4, n^2 + 1)$ random examples in $\mathcal{H}$. Since each of these $M_0$ examples has probability less than $\delta / 4 M_0$ of being in $\hat{r}$, we have confidence $1 - \delta / 4$ that none of these examples is in $\hat{r}$. Thus we can in fact find a set of $w_{ij}$ as in the proof of Theorem 2, and by Theorem 1 (with confidence $1 - \delta / 4$), this $w$ correctly classifies a fraction $1 - \epsilon$ of examples drawn from $D|_\mathcal{H}$. Now our hypothesis function is $x$ is positive if and only if $x \in \mathcal{H}$ and $\sum_{ij} x_i x_j > 0$. With confidence $1 - \delta / 8$ this correctly classifies all but a fraction much less than $\epsilon$ of points in $\mathcal{H}$ and with confidence $1 - 7\delta / 8$ it correctly classifies all but a fraction $\epsilon$ of points in $\mathcal{H}$. Q.E.D.
4 Some Remarks on Robustness

The algorithm we gave in Section 2 required, for proof of convergence, that the distribution be inversion symmetric. It also used a large number $O(n^3/\epsilon^2)$ examples, in spite of the fact that only $O(n/\epsilon)$ examples are known to be necessary for learning the class $\mathcal{N}$ and only $O(n^2/\epsilon)$ examples are necessary for training hypothesis nets with one hidden linear threshold and one hidden quadratic threshold unit. Both of these restrictions arose so that we could be sure of obtaining a large set of examples that is not in $\mathcal{F}$, and that are therefore known to be quadratically separable.

We proposed using Karmarkar's algorithm for the linear programming steps. This allowed us to guarantee convergence in polynomial time. However, if we had an algorithm that was able to find a near optimal linear separator, for a set of examples that is only approximately linearly separable, this might be much more effective in practice. For example, if in step 3 the set of examples we had was not exactly linearly separable, but instead contained by mistake either a few examples from $\mathcal{F}$ or a few noisy examples, we might still find an $\epsilon$ accurate classifier by using a robust method for searching for a near optimal linear separator. This would, for example, allow us to use fewer examples or to tolerate some variation from inversion symmetry.

Note that our method will (provably) work for a somewhat broader range of cases than strictly inversion symmetric distributions. For example, if we can write $D = D_1 + D_2$, where $D_1(x) = D_1(-x) \geq 0$ everywhere and $D_2(x) \geq D_2(-x)$ for all $x \in \mathbb{R}$, our proof goes through essentially without modification. ($D_2$ need not be positive definite.)

5 Learning with Membership Queries

The only use we made of the inversion symmetry of $D$ in the proof of Theorem 3 was to obtain a large set of examples that we had confidence was not in $\mathcal{F}$. The problem of isolating a set of examples not in $\mathcal{F}$ becomes trivial if we allow membership queries, since an example $y$ is in $\mathcal{F}$ if and only if $-y$ is a positive example. We may thus easily modify the algorithm using queries by finding a set $S_\mathcal{F}$ containing examples in $\mathcal{F}$ and a set $S_{\mathcal{F}}$ containing examples not in $\mathcal{F}$ (where membership in one or the other of these is easily established by query). We then find a half space $h$ containing all examples in $S_\mathcal{F}$ and correctly classify all examples in $S_{\mathcal{F}}$ by the method of Theorem 2. A straightforward analysis establishes (1) how many examples we need for sufficient confidence, relying on Theorem 1.

---

7 We will report elsewhere on a new, apparently very effective heuristic for finding near optimal linear separators.
Polynomial Time Algorithm

and (2) that we can with confidence acquire these examples rapidly. The algorithm is as follows.

1. Call examples, and for all negative examples \( y \), query whether \(-y\) is negative or positive. Accumulate in a set \( S_f \) all examples found such that \( y \) is negative but \(-y\) is positive. Accumulate in set \( S_f \), all examples not in \( S_f \). Continue until either (a) we have \( M_f = M(\epsilon/2, \delta/2, n + 1) \) examples in \( S_f \) and we have \( M = M(\epsilon/2, \delta/2, n^2 + 1) \) examples not in \( S_f \); or (b) one of the following happens. (b1) If in our first \( M_{\text{cut}} = \max\{4M_f/\epsilon, 16\epsilon^{-1}\ln(2\delta^{-1}), 2M_f/\epsilon\} \) examples we have not obtained \( M_f \) in \( S_f \), proceed to step 5. (b2) If instead we do not find \( M \) examples in \( S_f \) in these first \( M_{\text{cut}} \) calls proceed to step 6.

2. Find a half space \( h \) bounded by a hyperplane through the origin such that all examples in \( S_f \) are in \( h \) and all positive examples are in \( h \). \( h \) can be found, for example, using Karmarkar’s algorithm.

3. Find a \( w \in \mathbb{R}^n \) s.t. \( \sum_{ij} w_{ij} x_i x_j > 0 \) for any positive example \( x \in \bar{S}_f \) and \( \sum_{ij} w_{ij} x_i x_j < 0 \) for any negative example in \( S_f \). \( w \) can be found, for example, by Karmarkar’s algorithm.

4. Output hypothesis \( g: x \) is positive if and only if \( x \in h \) and \( \sum_{ij} w_{ij} x_i x_j > 0 \); and halt.

5. We conclude with confidence \( 1 - \delta/2 \) that fewer than a fraction \( \epsilon/2 \) of examples lie in \( r \). Thus we simply follow step 3; output hypothesis \( g: x \) is positive if and only if \( \sum_{ij} w_{ij} x_i x_j > 0 \); and halt.

6. We conclude with confidence \( 1 - \delta/2 \) that fewer than a fraction \( \epsilon/2 \) of examples are positive and thus simply output hypothesis: \( g(x) = 0 \) (i.e., classify all examples as negative) and halt.

**Theorem 4.** The class \( F \) of intersections of two half spaces is learnable from examples and membership queries.

**Proof.** It is easy to see with confidence \( 1 - \delta/2 \), as in the proof of Theorem 3, that if 1(b1) occurs, then the probability of finding an example in \( r \) is less than \( \epsilon/2 \). Thus we are allowed to neglect all examples in \( r \), provided that we correctly classify a fraction \( 1 - \epsilon/2 \) of all other examples. We then go to step 5, and find a quadratic classifier for the more than \( M \) examples we have in \( S_f \). By Theorem 1, as in the proof of Theorem 2, this gives us with confidence \( 1 - \delta/2 \) a classifier that is \( \epsilon/2 \) accurate. Thus overall if step (b1) is realized, we have confidence at least \( 1 - \delta \) of producing an \( \epsilon \)-accurate hypothesis.

Likewise, if (b2) occurs, then with confidence \( 1 - \delta/2 \) at most a fraction \( 1 - \epsilon/2 \) of examples are not in \( r \), and thus at most a fraction \( \epsilon/2 \) of examples are positive, and we are justified in hypothesizing all examples are negative, which we do in step 6.
Assume now we reach step 2. The half space $h$ found in step 2 contains, with confidence $1 - \delta/2$, all but a possible $\epsilon/2$ of the probability for examples in $\hat{r}$, by Theorem 1. The quadratic separator found in step 3 correctly classifies, by Theorem 1, with confidence $1 - \delta/2$, all but a fraction $1 - \epsilon/2$ of the measure not in $\hat{r}$. Note that there is no possibility that the set $\hat{S}_r$ will fail to be quadratically separable since it has no members in $\hat{r}$. This is in contrast to the situation in the proof of Theorem 3, where we had only statistical assurance of quadratic separability.

Putting this all together, we correctly classify with confidence $1 - \delta/2$ all but a fraction $\epsilon/2$ of the measure in $\hat{r}$, and with equal confidence all but the same fraction of measure not in $\hat{r}$. It is easy to see that, if we use some polynomial time algorithm such as Karmarkar's for the linear programming steps, that this algorithm converges in polynomial time. Q.E.D.

6 Concluding Remarks

The task of learning an intersection of two half spaces, or of learning functions described by feedforward nets with two hidden units, is hard and interesting because a Credit Assignment problem apparently must be solved. We appear to have avoided this problem. We have solved the task provided the boundary planes go through the origin and provided either membership queries are allowed or one restricts to inversion symmetric distributions.

This result suggests limited optimism that fast algorithms can be found for learning interesting classes of functions such as feedforward nets with a layer of hidden units. Hopefully the methods used here can be extended to interesting open questions such as: Can one learn an intersection of half spaces in the full distribution independent model, without membership queries? Can one find robust learning algorithms for larger nets, if one is willing to assume some restrictions on the distribution? We report elsewhere (Baum 1990b) on different algorithms that use queries to learn some larger nets as well as intersections of $k$ half spaces, in time polynomial in $n$ and $k$.

It is perhaps interesting that our algorithm produces as output function a net with two different types of hidden units: one linear threshold and the other quadratic threshold. It has frequently been remarked that a shortcoming of neural net models is that they typically involve only one type of neuron, whereas biological brains use many different types of neurons. In the current case we observe a synergy in which the

---

8Extension of our results to half spaces bounded by inhomogeneous planes is a subcase, since the inversion symmetry we use is defined relative to a point of intersection of the planes.

9Extension to intersections of more than two half spaces is an interesting subcase.
Polynomial Time Algorithm

A linear neuron is used to pick out only one of the two parabolic lobes that are naturally associated with a quadratic neuron. For this reason we are able readily to approximate a region defined as the intersection of two half spaces. It seems reasonable to hope that neural nets using mixtures of linear threshold and more complex neurons can be useful in other contexts.

Acknowledgments


References

Baum, E. B. 1990b. Neural net algorithms that learn in polynomial time from examples and queries. IEEE Transactions in Neural Networks, in press.


Received 16 April 90; accepted 3 July 90.